Respuesta Tarea 8, Claudia Acosta Díaz y Axel Báez Ochoa (Fecha de entrega 12/10/2020)

Parte 1:
La implementación del punto 1 está en el archivo heap.c de la carpeta Axel-Claudia en la rama Axel. Se prueba en test1.c y en specs.c. Se hizo una implementación que considera una variable char type en la estructura de montículo para especificar si es un montículo max (M) o min (m). El punto 2 que es la complejidad, está escrita en el código.

Parte 2:

Punto 1: El código para la mediana está en el archivo median.c y se prueba en test2.c y en specs.c, todo está en la carpeta Axel-Claudia en la rama Axel.

Punto 2: Se debe probar que el algoritmo propuesto calcula la mediana correctamente.

Supongamos que se tienen n datos y que su mediana es m. Además, siempre se tiene que el montículo que guarda la parte inferior de los datos es tipo max y el que guarda la parte superior de los datos es montículo min.

Si n es par y se introduce un nuevo dato. Por la forma del algoritmo, si n es par, se tendrá que max_heap y min_heap tendrán la misma cantidad de datos. Por tanto, se introducirá el nuevo dato en uno de los dos montículos según sea mayor o menor que la mediana. El montículo al cual se introduce el dato se arregla acorde a las propiedades de montículo. Es decir, si es max, quedará en la raíz el valor máximo de los datos y si es min, el valor mínimo. Luego, el montículo donde se insertó el dato tendrá n+1 valores, por tanto, su raíz, si se ordenara toda la colección, sería el número del medio y por tanto la mediana, que es como lo toma el algoritmo.

Si n es impar, entonces uno de los dos montículos max_heap o min_heap tiene un dato más que el otro. En ese caso, el algoritmo compara con la mediana y decide donde pondrá el dato. Si lo pone en el de mayor tamaño, entonces quita uno de ese montículo y lo pasa al otro. Si lo pone en el de menor tamaño, entonces ya quedan equiparados. Cada vez que inserta, reorganiza los montículos para que guarden su propiedad de ser montículos max o min según corresponda. Después de este proceso, siempre quedan los montículos con igual número de datos y de manera que si se ordenaran los n+1 (n+1 es par) datos en un arreglo, los dos datos del centro serán justamente el mayor de la parte izquierda (los menores) y el menor de la parte derecha (los mayores), que son justamente las raíces de los montículos, por tanto, como el algoritmo toma el promedio de esos dos datos, estaría calculando la mediana nueva correctamente.

Punto 3: Se debe ver la complejidad del algoritmo.

En este problema, si se insertan n elementos en total, se estarán insertando $\frac{n}{2}$ o $\frac{n}{2}+1$ a cada montículo max_heap o min_heap, dependiendo de si n es par o impar. En cualquier caso, como se vio en la parte uno, esto tomaría complejidad $O(log_3(n/2))$ que es $O(log_3(n))$. Si lo vemos paso a paso, al insertar el primer valor, sería $O(log(1))$, para el segundo valor $O(log(1))$ también pues se pone en el montículo vacío. Para el tercer valor y el cuarto, sería $O(log(2))$, luego, para el quinto y sexto, $O(log(3))$ y así sucesivamente hasta para n y n-1 que sería $O(log(n/2))$. Si se mira esto, siempre tendremos que $log(i) < log(n/2)$ si $i < n/2$. Por tanto,

$$2 * (log(1) + log(2) + ... + log(n/2)) \leq 2 * (n/2)(log(n/2)) \in O(nlog(n)) \quad (1)$$

Por tanto el algoritmo es $O(nlog(n))$. NOTA: Todos los logaritmos son en base 3.